Renault SAS Nissan Peugeot Citroën Automobile	Reference : **0001-2** Version **1.0**	Publication date **2009/01/29** Page 1/15

Standard ECU reprogramming
Part 2 - Ecu-tool programming interfaces description

Comment:
This document describes interfaces between a programming tool and the boot-loader of an ECU.
Those interfaces are the one related to programming procedure.
Interfaces are an implementation of ISO 14229-1 Unified Diagnostic Services.

AUTHORS(S)	
Name : Gilles Michard (Renault SAS) Cédric Meunier (Peugeot Citroën Automobile)	

1 Revision summary

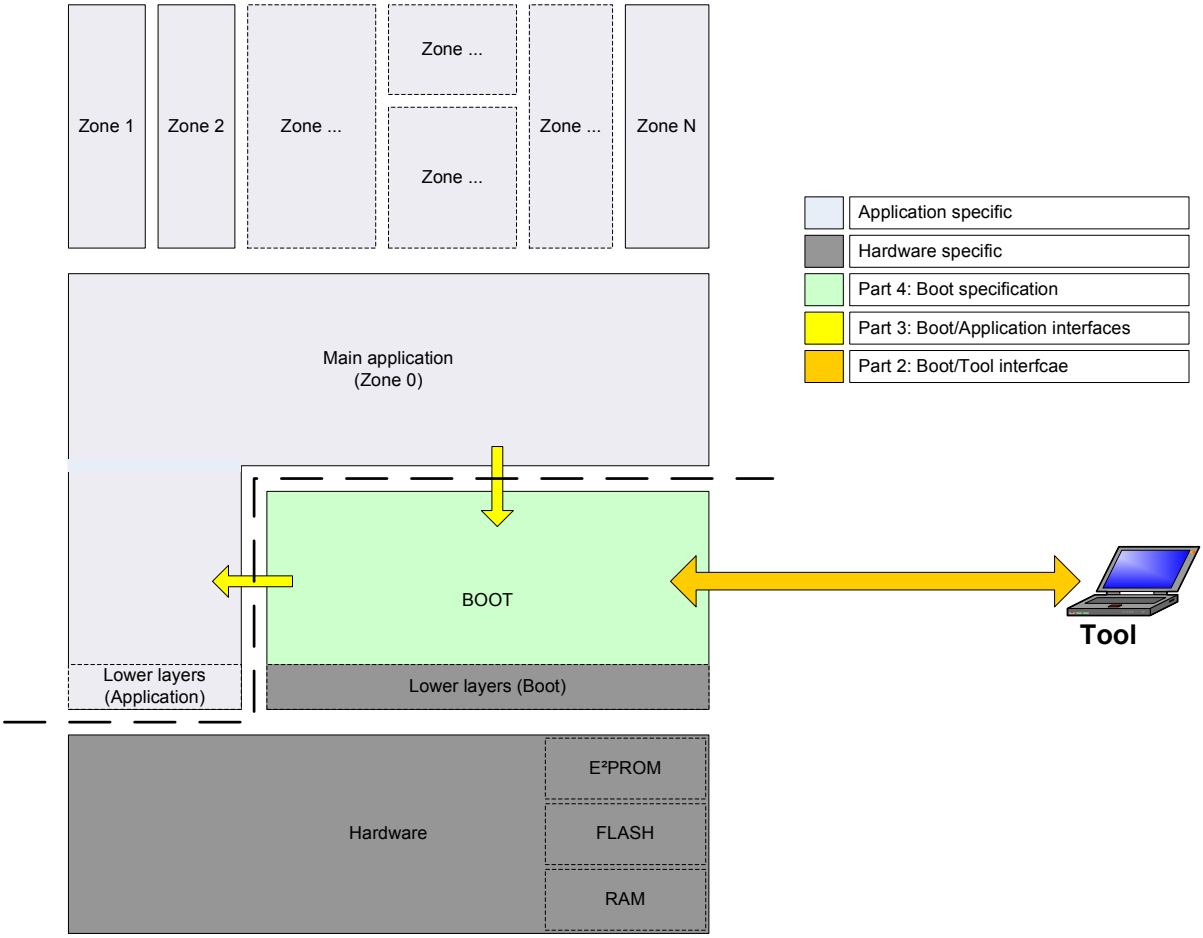
Revision	Date	Modified paragraphs and kind of modification
1.0	January 29th, 2009	First edition

2 Content

3 PURPOSE

This document is a part of Standard ECU reprogramming specification package. This package is divided into five parts, consistent to each others.

- Part 1: General description
- Part 2: Boot-tool programming interfaces description
- Part 3: Boot-loader-Application interface description
- Part 4: Boot-loader mechanisms
- Part 5: Conformance test



Original documents will be freely available on "Internet Archive" website:
http://www.archive.org/

The Standard ECU reprogramming package contains specification on boot, how it interfere with application (application reprogramming and launch), with tool (how to reprogram an application, conformity test) and means to test the boot.

This document describes interfaces between a programming tool and the boot-loader of an ECU.
Those interfaces are the one related to programming procedure.
Interfaces are an implementation of ISO 14229-1 Unified Diagnostic Services.

Boot-loader interfaces are:

	Name	**Comment**
Boot/Application		
Boot/Tool	BOOT_PROG_SESSION	
	BOOT_SA_UNLOCK_REQ	
	BOOT_WRITE_DIGEST	
	BOOT_REQ_DOWNLOAD	
	BOOT_TRANSFER_DATA	
	BOOT_RUNDTCTEST	
	BOOT_GET_LOGICALBLOCK_INFO	
Boot/Lower layer		

4 APPLICABLE DOCUMENTS

4.1 References documents

Title	Ref.	Rev.
[1] Standard ECU reprogramming - Part 4 - Boot-loader mechanisms	0001-4	1.0

4.2 Norms and Procedures

Title	Ref.	Rev.
[2] Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements	ISO 14229-1	2006
[3] Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part 3:Implementation of unified diagnostic services (UDS on CAN)	ISO 15765-3	2004
[4] Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 6: Diagnostic trouble code definitions	ISO 15031-6	2005

5 TERMINOLOGY

5.1 Glossary

Segment	:	A segment is a in memory contiguous part of data.
Logical Block	:	A logical block is a block of consistent data that can be unitarily be reprogrammed (e.g. calibration or software module). A logical block is build of one or more segments.
Unlocking fingerprint	:	This is a data that represent the entity for which securityAccess has been allowed..
Digest	:	Result of a cryptographic hash function. It represents the calculated signature of a data set.
Scratchpad	:	Volatile memory part that can be programmed without any security check.
Memory handler	:	Procedure that can access (read/write) to Logical Block
Public Key	:	Key stored into ECU used for data encryption using asymmetrical cryptographic algorithms.
Secured Session Identifier	:	Data that identify the session (tool type, date, tool owner, ….) for which the ECU have been unlocked

5.2 Abbreviations and acronyms

NRC : Negative Response Code. See document [2]

5.3 Conventions

5.3.1 Requirements presentation

Requirements are presented in the form of a table containing following information:
- First column : Requirement identifier (see below the applicable numbering method)
- Second column : Requirement Description

A requirement can be followed by two row, a rationale and a comment that can help the reader to understand the requirement.

STD-PRG2-TS.nnnn(V)	Requirement Description
Rationale	*Rationale on the above requirement.*
Comment	*Comment on the above requirement*

5.3.2 Requirement identifiers

For an easy identification of the requirement scope, the following method has been adopted: STD-PRG2-TS.nnnn(V)
where:

STD	:	For standard requirement
PRG	:	For programming related requirement.
2	:	For part 2 related requirement.
TS	:	For technical specifications
nnnn	:	Requirement Identifier number (from 0000 à 9999)
V	:	Requirement version number

6 REQUIREMENTS

6.1 General

STD-PRG2-TS.0001(1)	Communication with the boot-loader is ISO UDS compliant. See document [2]
Rationale	*-*
Comment	*-*

6.2 BOOT_RUNDTCTEST

STD-PRG2-TS.0100(1)	The UDS service used for BOOT_RUNDTCTEST is RoutineControl.
Rationale	*-*
Comment	*-*

6.2.1 Request

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**RoutineControl**	**RC**	**0x31**	M.
2	7	SuppresPosRspMsgIndicationBit	SPRMIB	0b0 - 0b1	M.
2	6 à 0	RoutineControlType	RCTP	0x00 - 0x7F	M.
3 4	-	RoutineIdentifier : MSB LSB	RI	 0x00 - 0xFF 0x00 - 0xFF	M
5 … 7	-	RoutineControlOptionRecord : RoutineControlOption #1 … RoutineControlOption #3	RCEOR	 0x00 - 0xFF … 0x00 - 0xFF	C/U

STD-PRG2-TS.0101(1)	RCTP parameter is implemented as describes into document ISO UDS [2]. Its value is always StartRoutine 0x01
Rationale	*In order to simplify the boot-loader software, this routine is implemented as synchronous routine.*
Comment	*If the routine needs more than P2_server to be executed (see ISO 15765-3 [3] document), NRC 0x78 (see ISO 14229-1 [2] document) will be used.*

STD-PRG2-TS.0102(1)	RI parameter is equal to RDTCT (0XFF06)
Rationale	*-*
Comment	*The value is subject to change depending on ISO standardization.*

Proposed change request to ISO 14229-1 Table F.1 — routineIdentifier definition

FF06	**runDTCTest** This value shall be used to run the test for the requested DTC and update accordingly the DTC status.	M	RDTCT_

STD-PRG2-TS.0103(1)	RCEOR RoutineControlOption #1 to #3 content is the DTC to be tested. See ISO 15031-6 [4] document.
Rationale	*-*
Comment	*This routine is used to force the DTC to be tested.*

6.2.2 Response

Positive response:

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**RoutineControl PR**	**RCPR**	**0x71**	Obl.
2	-	RCTP and SPRMIB of the request	RCPR	0x00 - 0xFF	Obl.
3-4	-	RI	RI	RDTCT	Obl.
5-6	-	RoutineStatusRecord : RoutineStatus #1 : RoutineStatus #2 :	RSR	0x00 0x00 - 0xFF	Obl.

STD-PRG2-TS.0104(1)	RSR RoutineStatus #1 is fixed to 0
Rationale	*This is the usage and is about to be normalized. See ISO 26021-2 routines.*
Comment	*-*

STD-PRG2-TS.0105(1)	RSR RoutineStatus #2 is equal to the updated status of the DTC sent into RCEOR request parameter.
Rationale	*-Avoid checking DTC status with service readDTCInformation.*
Comment	*This positive response implies that the "notTested" DTC status bit is now mandatory set to 0. Otherwise, the appropriate negative response is sent.*

As the routine is a synchronous routine, no other positive response can be sent by the server.

6.3 BOOT_PROG_SESSION

6.3.1 Request

STD-PRG2-TS.0200(1)	The UDS service used for BOOT_PROG_SESSION is DiagnosticSessionControl.
Rationale	*-*
Comment	*-*

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**DiagnosticSessionControl**	**DSC**	**0x10**	Obl.
2	7	SuppresPosRspMsgIndicationBit	SPRMIB	0b0 - 0b1	Obl.
2	6 à 0	DiagnosticSessionType	DS	0x00-0x7F	Obl.

STD-PRG2-TS.0201(1)	The UDS service used for BOOT_PROG_SESSION is DiagnosticSessionControl.
Rationale	*-*
Comment	*-*

STD-PRG2-TS.0202(1)	DS is equal to 0x02 for programmingSession, according to ISO 14229-1 UDS norm [2]
Rationale	*-*
Comment	*-*

6.3.2 Response

The positive response of BOOT_PROG_SESSION is conform to ISO 14229-1 UDS [2] document.

STD-PRG2-TS.0203(1)	sessionParameterRecord for positive response of BOOT_PROG_SESSION indicates 50ms for P2max timing and 5s for P2*max. sessionParameterRecord []= Byte #1 = 0x00 Byte #2 = 0x32

	Byte #3 = 0x13 Byte #4 = 0x88
Rationale	*-*
Comment	*This format is today specified by ISO 15765-3 only, but is expected to be included in the next revision of UDS, independently of communication medium.* *The values should be the same as in the default session*

6.4 BOOT_REQ_DOWNLOAD

STD-PRG2-TS.0300(1)	The UDS service used for BOOT_REQ_DOWNLOAD is RequestDownload.
Rationale	*-*
Comment	*-*

6.4.1 Request

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**RequestDwnload**	**RD**	**0x34**	M.
2	-	dataFormatIdentifier	DFI_	0x00-0xFF	M.
3	-	addressLengthFormatIdentifier	ALFID	0x00-0xFF	M.
4 … m		memoryAddress	MA_	0x00-0xFF … 0x00-0xFF	M
m+1 … n		memorySize	MS_	0x00-0xFF … 0x00-0xFF	M

STD-PRG2-TS.0301(1)	memoryAddress contains the starts of a memory Segment.
Rationale	*-*
Comment	*-*

STD-PRG2-TS.0302(1)	memorySize contains the size of the memory Segment that starts at memoryAddress address.
Rationale	*-*
Comment	*-*

addressLengthFormatIdentifier is filled accordingly to ISO 14229-1 UDS [2] document.

DataFormatIdentifier specifies the options for encryption/compression. The 0 value means no encryption and no compression and must be supported. Additional support of the inflate algorithm (IETF RFC 1951 DEFLATE Compressed Data Format) in the boot-loader allows a typical 50% gain on software download size.

6.4.2 Response

No specific constraint on RequestDownload positive response. However the maxNumberOfBlockLength in the response which specifies the maximum size used for the TransferData service must not be too small, preventing efficient use of the transport layer.

6.5 BOOT_TRANSFER_DATA

STD-PRG2-TS.0400(1)	The UDS services used for BOOT_TRANSFER_DATA are TransferData.and TransferExit.
Rationale	*-*
Comment	*-*

6.5.1 Requests

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**TransferData**	**TD**	**0x36**	M.
2	-	blockSequenceCounter	BSC	0x00-0xFF	M.
3 … m		transferRequestParameterRecord	TRPR_	0x00-0xFF … 0x00-0xFF	M

blockSequenceCounter is filled accordingly to ISO 14229-1 UDS [2] document.

transferRequestParameterRecord is filled accordingly to ISO 14229-1 UDS [2] document.

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**RequestTransferExit**	**RTE**	**0x37**	M.

STD-PRG2-TS.0401(1)	No parameters is implemented for RequestTransferExit service.
Rationale	*-*
Comment	*-*

6.5.2 Response

No specific constraint on transferData positive response.

6.6 BOOT_SA_UNLOCK_REQ

STD-PRG2-TS.0500(1)	The UDS services used for BOOT_SA_UNLOCK_REQ is SecurityAccess.
Rationale	*-*
Comment	*-*

STD-PRG2-TS.0501(1)	The securityAccessType values used for BOOT_SA_UNLOCK_REQ, are 1 and 2.
Rationale	*-*
Comment	*-*

6.7 BOOT_WRITE_DIGEST

STD-PRG2-TS.0600(1)	The UDS services used for BOOT_WRITE_DIGEST is WriteDataByIdentifier
Rationale	*-*
Comment	*-*

6.7.1 Request

Byte	Bit	Service name and parameter	Mnemonic	Value	Cvt
1	**-**	**WriteDataByIdentifier**	**WDBI**	**0x2E**	M.
2 3	-	DataIdentifier MSB LSB	DID	0x00-0xFF 0x00-0xFF	M.
4 … m		DataRecord	DREC	0x00-0xFF … 0x00-0xFF	M

STD-PRG2-TS.0601(1)	Logical Block DataIdentifier is the same value as Logical Block base DTC
Rationale	*-*
Comment	*As an example, if Logical Block DTC is 0x123451 (base DTC is 0x1234, failure type 0x51, programming failure/not programmed), Logical Block*

	DataIdentifier is also 0x1234.

STD-PRG2-TS.0602(1)	DataRecord contains Logical Block's digest.
Rationale	*-*
Comment	*If hash algorithm is SHA-1, DataRecord length is 20.*

6.7.2 Response

No specific constraint on WriteDataByIdentifier positive response.

6.8 BOOT_GET_LOGICALBLOCK_INFO

6.8.1 Request

STD-PRG2-TS.0700(1)	The UDS service used for BOOT_GET_LOGICALBLOCK_INFO is ReadDTCInformation..
Rationale	*-*
Comment	*-*

STD-PRG2-TS.0701(1)	The UDS ReadDTCInformation sub-function used for BOOT_GET_LOGICALBLOCK_INFO is reportDTCFaultDetectionCounter..
Rationale	*-*
Comment	*-*

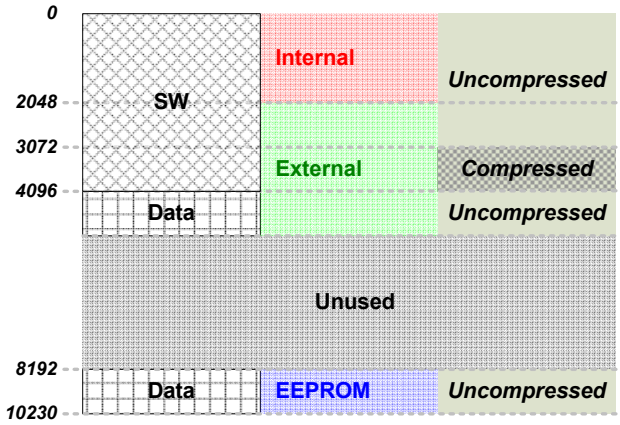
6.8.2 Response

STD-PRG2-TS.0702(1)	For a logical block with associated counter(s), the number of remaining operations is equal to 126 – DTCFaultDetectionCounter for its associated DTC with fault type "general memory failure" (0x42). If this number is more that 125, that DTC is not reported into the response. When the reported number is 127 (-1 remaining operations), the DTC is already failed.
Rationale	*According to ISO 14229-1 UDS [2] norm (see table 249), DTCFaultDetectionCounter must be a number between 1 and 127. DTCFaultDetectionCounter equals to 127 represents a test result of "failed".*
Comment	*This data is only available if less than 126 programming operations remain.* *A value of 126 means the memory can no more be safely updated*

7 ANNEXES

Dataset's covers entirely memory mapping of logical blocks.
Digests can be re-calculated.

7.1 Memory mapping example:



7.2 Example of data file content

These examples show a data model containing information that a tool will need to effectively program an ECU. The format for delivery of flash content from the system supplier to the vehicle manufacturer should instead be based on ISO 22901-1 Open Diagnostic Data Exchange.

```
<?xml version="1.0" encoding="utf-8" ?>
<boot name="demo">
  <hashFunction name="sha1"/>
  <compressionMethod name="none" value="0"/>
  <compressionMethod name="deflate" value="1"/>
  <encryptingMethod name="none" value="0"/>
  <alfid address="4" size="2"/>
  <sector at="0" size="2048" physicalMemory="internal" logicalBlock="software"/>
  <sector at="2048" size="1024" physicalMemory="external" logicalBlock="software"/>
  <sector at="3072" size="1024" physicalMemory="external" logicalBlock="software"/>
  <sector at="4096" size="2048" physicalMemory="external" logicalBlock="data"/>
  <sector at="8192" size="2048" physicalMemory="eeprom" logicalBlock="data"/>
  <physicalMemory name="internal" erasedPattern="00000000" maxErasing="30">
    <handler>
      <download compressionMethod="none" encryptingMethod="none" at="65536"
size="1024">BBBB</download>
    </handler>
  </physicalMemory>
  <physicalMemory name="external" erasedPattern="FFFF">
    <handler>
      <download compressionMethod="none" encryptingMethod="none" at="65536"
size="1024">CCCC</download>
    </handler>
  </physicalMemory>
  <physicalMemory name="eeprom" erasedPattern ="">
    <handler>
      <download compressionMethod="none" encryptingMethod="none" at="65536"
size="1024">DDDD</download>
    </handler>
  </physicalMemory>
  <logicalBlock name="software" id="EE40">
    <dataset name="soft1" digest="XXXXXXXX">
      <download physicalMemory="internal" compressionMethod="none" encryptingMethod="none"
at="0" size="2048">AAAA</download>
      <download physicalMemory="external" compressionMethod="none" encryptingMethod="none"
at="2048" size="1024">AAAA</download>
      <download physicalMemory="external" compressionMethod="deflate" encryptingMethod="none"
at="3072" size="1024">AAAA</download>
    </dataset>
```

```
    <dataset name="soft2" digest="YYYYYYYY">
      <download physicalMemory="internal" compressionMethod="none" encryptingMethod="none"
at="0" size="2048">BBBB</download>
      <download physicalMemory="external" compressionMethod="none" encryptingMethod="none"
at="2048" size="1024">BBBB</download>
      <download physicalMemory="external" compressionMethod="deflate" encryptingMethod="none"
at="3072" size="1024">BBBB</download>
    </dataset>
  </logicalBlock>
  <logicalBlock name="data" id="EE41">
    <dataset name="data1" digest="AAAAAAAA">
      <download physicalMemory="external" compressionMethod="none" encryptingMethod="none"
at="4096" size="2048">AAAA</download>
      <download physicalMemory="eeprom" compressionMethod="none" encryptingMethod="none"
at="8192" size="2048">AAAA</download>
    </dataset>
    <dataset name="data2" digest="AAAAAAAB">
      <download physicalMemory="external" compressionMethod="none" encryptingMethod="none"
at="4096" size="2048">AAAB</download>
      <download physicalMemory="eeprom" compressionMethod="none" encryptingMethod="none"
at="8192" size="2048">AAAA</download>
    </dataset>
  </logicalBlock>

</boot>
```

7.3 Data file schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema          attributeFormDefault="unqualified"          elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="download">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="compressionMethod" type="xs:string" use="required" />
        <xs:attribute name="encryptingMethod" type="xs:string" use="required" />
        <xs:attribute name="at" type="xs:long" use="required" />
        <xs:attribute name="size" type="xs:long" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name ="nibble">
    <xs:restriction base ="xs:int">
      <xs:minInclusive value ="0"/>
      <xs:maxInclusive value ="15"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="boot">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="hashFunction">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="compressionMethod">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="value" type="nibble" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="encryptingMethod">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="value" type="nibble" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="alfid">
          <xs:complexType>
            <xs:attribute name="address" type="nibble" use="required" />
            <xs:attribute name="size" type="nibble" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="sector">
          <xs:complexType>
            <xs:attribute name="at" type="xs:long" use="required" />
            <xs:attribute name="size" type="xs:long" use="required" />
            <xs:attribute name="physicalMemory" type="xs:string" use="required" />
            <xs:attribute name="logicalBlock" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="physicalMemory">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="handler">
                <xs:complexType>
```

```
                  <xs:sequence>
                    <xs:element name="download" type="download"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="erasedPattern" type="xs:hexBinary"  use="required"/>
            <xs:attribute name="maxErasing" type="xs:int" />
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="logicalBlock">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="dataset">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="download">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="download">
                            <xs:attribute name="physicalMemory" type="xs:string"
use="required" />
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="name" type="xs:string" use="required" />
                  <xs:attribute name="digest" type="xs:base64Binary" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="id" type="xs:hexBinary" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```